Blind identification of an unknown interleaved convolutional code

Audrey Tixier
Inria, Le Chesnay BP 105, 78153 Cédex

1 Introduction

Blind identification of an unknown code from the observation of noisy codewords. We address here a problem related to cryptanalysis and data security where an observer wants to extract information from a noisy data stream where the error correcting code which is used is unknown. Basically an observer has here several noisy codewords originating from an unknown code and wants to recover the unknown code and decode it in order to recover the whole information contained in these codewords. Generally this problem is solved by making assumptions on the code which is used in this scenario (convolutional code, LDPC code, turbo-code, concatenated code, etc.). This is called the code reconstruction problem or blind identification of a code problem in the literature. This problem arises for instance in a non-cooperative context where observing a binary sequence originating from an unknown communication system naturally leads to such a problem, for more details see the introduction of [1]. It also arises in the design of cognitive receivers which are able to cope with a great variety of error correcting codes [1] or in the study of DNA sequences when looking for possible error correcting codes in the genetic code [2]. This problem has a long history: it has been addressed for a variety of codes, linear codes [3–6], cyclic codes [7–10], LDPC codes [5, 6], convolutional codes [1, 11–26], turbo-codes [27–37], BCH codes [38–41], Reed-Solomon codes [42–44] and Reed-Muller codes [45].

We focus here on the problem of reconstructing an unknown code when an interleaver is applied after encoding. Recall that this is a commonly used technique to correct burst errors since it provides some sort of time diversity in the coded sequence. For instance when the code is a convolutional code,

this allows to spread the burst errors in remote locations and convolutional decoding performs much better. This problem has been already addressed in a series of papers [46–50]. All these papers assume that the interleaver is structured (a convolutional interleaver [46–49] or an helical scan interleaver [50]). It should be said here that the methods used in these papers make heavily use of the particular structure of the interleaved that is considered and some of these methods are highly sensitive to noise [47, 49] or do not apply when there is noise [46, 48].

In this paper, we will focus on the case where the code is a convolutional code and where the interleaver is a block interleaver. This pair is used in several standards, for example, in 802.11n, 802.16e, 802.22, GMR-1 and Wimedia. In practice the block interleaver is structured but this structure is not always the same, so to reconstruct the interleaver in all cases we will assume that the interleaver has no particular structure, it is chosen randomly among all possible permutations. Our problem can be formalized as follows.

Blind identification of an unknown interleaved convolutional code: statement of the problem, hypotheses and notations. The encoding process which is studied in this paper is described in Figure 1 and consists in taking an information word of length mk and feeding it into an (n, k) convolutional encoder to produce a codeword $\boldsymbol{x}$ of length $N = mn$. We denote by $\mathcal{C}$ the set of codewords obtained by this convolutional code (in other words this is a convolutional code truncated in its first N entries). The codeword $\boldsymbol{x}$ is then permuted by a fixed block interleaver π of length N, we denote by $\boldsymbol{y}$ the interleaved codeword. $\boldsymbol{y}$ is then sent through a binary symmetric channel of crossover probability p. At the output of the channel we observe the noisy interleaved codeword $\boldsymbol{z}$.

The blind identification process consists in observing M noisy interleaved codewords $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^M$ to recover the convolutional code $\mathcal{C}$ and the block interleaver π. The codeword and interleaved codeword associated to $\boldsymbol{z}^i$ are respectively denoted by $\boldsymbol{x}^i$ and $\boldsymbol{y}^i$. We also denote by $\mathcal{C}_\pi$ the code $\mathcal{C}$ interleaved by π ($\boldsymbol{y}^1, \ldots, \boldsymbol{y}^M$ belong to it). We will assume that the length N of the interleaver is known. It can obtained through the techniques given in [51–54] and recovering this length can now be considered to be a solved problem. However, contrarily to [46–50] we will make no assumption on the interleaver: it is chosen randomly among all permutations of size N. The convolutional code $\mathcal{C}$ is assumed to be unknown, its parameters n and k are also unknown.

Our contribution. In this paper, we reconstruct the block interleaver π

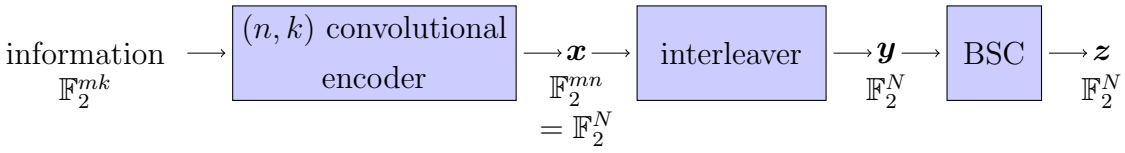

Figure 1: The communication scheme considered here.

and we recover the convolutional code $\mathcal{C}$. For this, we search for the dual code of $\mathcal{C}_\pi$. This code is recovered by known techniques [6] which are adapted to recover parity-check equations of low weight given noisy codewords. Once $\mathcal{C}_\pi$ is recovered, we classify the parity-check equations that have been obtained for $\mathcal{C}_\pi$ into groups. Each group contains parity-check equations that correspond to parity-check equations of $\mathcal{C}$ whose positions differ by a multiple of n. We will then focus on a specific group of equations and will be able to reconstruct the convolutional code and the unknown interleaver at the same time by introducing novel graph techniques in this setting.

By running some experimental tests we have been able to demonstrate the efficiency of this method. For example, for a convolutional code with parity-check equations of weight 6, after finding a set of the parity-check equations, we reconstruct an interleaver of length 8000 in less than ten seconds. The time for finding the interleaver once $\mathcal{C}_\pi$ has been recovered does not depend on the noise level, the noise only impacts searching the parity-check equations of $\mathcal{C}_\pi$. It should be added that the method used to recover these parity-check equations is more efficient than the methods calculating the rank of matrices, in particular when the data are noisy. This allows us to reconstruct efficiently the interleaver even for moderate noise levels.

2 Overview of the algorithm

Our reconstruction algorithm makes heavily use of two properties of the parity-check equations of an (n, k) convolutional code

(i) for small up to moderate constraint length (which is the case of all convolutional codes used in practice) the parity-check equations are of low weight;

(ii) with the exception of a few parity-check equations involving the first bits, shifts of a parity-check equations by a multiple of n are also parity-check equations of the convolutional code.

Interleaving does not destroy the first property but the second property is lost for $\mathcal{C}_\pi$. From now on we denote by t the "essential" minimum weight of parity-check equations of $\mathcal{C}$ (or $\mathcal{C}_\pi$). By essential minimum weight we mean here that we take the minimum of the weights that appear at least a linear (in N) number of times. We use this definition to discard parity-check equations that could involve the first bits of $\mathcal{C}$ and that could be of lower weight due to the zero initialization of the convolutional code. For the reconstruction we need a list $\mathcal{L}$ of these parity-check equations of weight t.

Our algorithm basically works as follows

1. We use the algorithm of [6] to find a list $\mathcal{L}$ of parity-check equations of weight t.

2. We classify the parity-check equations of $\mathcal{L}$ into disjoint groups $\mathcal{L}_1, \ldots, \mathcal{L}_r$ such that two parity-check equations fall into the same group if, and only if, they correspond to parity-check equations of $\mathcal{C}$ which are shifts of each other by a multiple of n.

3. Denote by $\mathcal{L}_1$ the group of parity-check equations of $\mathcal{L}$ which have the smallest intersection number. The intersection number of a parity-check equation $\mathcal{E}$ is the number of parity-check equations in $\mathcal{L}$ which have at least one position in common with $\mathcal{E}$. We use this group to recover one parity-check equation of $\mathcal{C}$ by graph theoretic considerations.

4. We use this parity-check equation of $\mathcal{C}$ to reorder the parity-check equations in $\mathcal{L}_1$. Note that the structure of the group $\mathcal{L}_1$ is such that these ℓ parity-check equations $\mathcal{E}_1, \ldots, \mathcal{E}_\ell$ correspond to ℓ parity-check equations $\mathcal{E}'_1, \ldots, \mathcal{E}'_\ell$ of $\mathcal{C}$ which are shifts of a multiple of n of each other. This reordering is done in such a way that $\mathcal{E}'_i$ is the shift by $n(i-1)$ of $\mathcal{E}'_1$.

5. This reordering of $\mathcal{L}_1$ is then used in the last step to recover π.

3 Notation

A parity check equation $\mathcal{E}$ will be denoted by the set of positions it involves, when we write $\mathcal{E} = \{e_1, \ldots, e_t\}$ we mean here that this parity-check equation involves the positions $\{e_1, \ldots, e_t\}$ of the code that is considered (which is generally clear from the context).

With set notation applying an interleaver π to code positions really amounts to transform a parity-check equation $\mathcal{E} = \{e_1, \ldots, e_t\}$ into a parity-check equation $\pi(\mathcal{E}) \stackrel{\text{def}}{=} \{\pi(e_1), \ldots, \pi(e_t)\}$.

4 The reconstruction algorithm in detail

4.1 Recovering parity-check equations of weight t

The first step consists in searching for a list $\mathcal{L}$ of parity-check equations of $\mathcal{C}_\pi$ of weight t. To obtain this list we apply the algorithm of [6]. This method allows us to find the parity-check equations even if the observed codewords are noisy.

4.2 Classifying parity-check equations into groups

We want to classify parity-check equations of $\mathcal{L}$ into disjoint groups $\mathcal{L}_1, \ldots, \mathcal{L}_r$ such that the parity-check equations in a group correspond to parity-check equations of $\mathcal{C}$ that are shifts of each other by a multiple of n. We say that these parity-check equations are of the same type.

Definition 1 (Type of a parity-check equation of $\mathcal{C}$)**.** *$\mathcal{E} = \{e_1, \ldots, e_t\}$ and $\mathcal{E}' = \{e'_1, \ldots, e'_t\}$ two parity-check equations of $\mathcal{C}$ are of the same type if $\mathcal{E}'$ is a shift by a multiple of n of $\mathcal{E}$. This means that there exists i such that $\{e_1, \ldots, e_t\} = \{e'_1 + in, \ldots, e'_t + in\}$. In such a case we write $\mathcal{E} \sim \mathcal{E}'$.*
All parity-check equations of the same type define an equivalence class.

Why classify ? We need to classify parity-check equations of $\mathcal{L}$ because our method uses the regularity of parity-check equations of the convolutional code: shifts of parity-check equations by a multiple of n are also parity-check equations of the convolutional code. A convolutional code can satisfy several types of parity-check equations of the same weight t.

Example 1. *The* $(2,1)$ *convolutional code which satisfies* $x_1+x_2+x_3+x_5+x_6+x_8=0$ *also satisfies its shifts:* $\forall i$, $x_{1+2i}+x_{2+2i}+x_{3+2i}+x_{5+2i}+x_{6+2i}=0$. *If we add two consecutive parity-check equations, we obtain another parity-check equation:* $x_{1+2i}+x_{2+2i}+x_{4+2i}+x_{6+2i}+x_{7+2i}+x_{10+2i}=0$. *This equation is verified for all integers* i. *So this code has at least two equivalence classes of parity-check equations: the first is represented by* $\mathcal{E}=\{1,2,3,5,6,8\}$ *and the second by* $\mathcal{E}'=\{1,2,4,6,7,10\}$. *The weight of all these parity-check equations is* 6. *In this case, when we search for parity-check equations of weight* 6 *of* $\mathcal{C}_\pi$ *we find equations corresponding to a mixture of these two types.*

If we had directly parity-check equations of $\mathcal{C}$ instead of parity-check equations of $\mathcal{C}_\pi$, then these different types of parity-check equation might get differentiated by their span:

Definition 2 (Span of a parity-chek equation)**.** *Let* $\mathcal{E}=\{e_1,\ldots,e_t\}$ *be a parity-check equation, its span* $s^{\mathcal{E}}$ *is defined by* $s^{\mathcal{E}}=\max_i(e_i)-\min_i(e_i)+1$. *In an equivalence class, all parity-check equations have the same span and we call this quantity the span of the equivalence class.*

Once interleaving this property is lost but the equivalence classes are always present:

Definition 3 (Type of a parity-check equation of $\mathcal{C}_\pi$)**.** *Two parity-check equations* $\mathcal{E}$ *and* $\mathcal{E}'$ *of* $\mathcal{C}_\pi$ *are of the same type if* $\pi^{-1}(\mathcal{E})\sim\pi^{-1}(\mathcal{E}')$.

How to classify? Even if we can not use the notion of the span of parity-check equations to classify the equations of $\mathcal{C}_\pi$, we will use the related notion of neighbourhood profile:

Definition 4 (Neighbourhood profile)**.** *Let* $\mathcal{E}\in\mathcal{L}$, *its neighbourhood profile* $\mathcal{P}^{\mathcal{E}}$ *is a vector of length* t: $\mathcal{P}^{\mathcal{E}}=(\mathcal{P}_1^{\mathcal{E}},\ldots,\mathcal{P}_t^{\mathcal{E}})$ *where* $\mathcal{P}_i^{\mathcal{E}}=\#\{\mathcal{E}'\in\mathcal{L}$ *such as* $|\mathcal{E}\cap\mathcal{E}'|=i\}$.

In other words, for a parity-check equation $\mathcal{E}$, $\mathcal{P}_i^{\mathcal{E}}$ is equal to the number of parity-check equations which have exactly i positions in common with $\mathcal{E}$. The number of parity-check equations with at least one position in common with $\mathcal{E}$ defines its intersection number:

Definition 5 (Intersection number)**.** *The intersection number* $\mathcal{I}^{\mathcal{E}}$ *of a parity-check equation* $\mathcal{E}\in\mathcal{L}$ *is equal to* $\mathcal{I}^{\mathcal{E}}=\sum_{i\leq t}\mathcal{P}_i^{\mathcal{E}}$.

Use profiles to determine the type of parity-check equations. The point of Definition 4 is that all parity-check equations of $\mathcal{C}$ of the same type have the same neighbourhood profile, whereas two equations of two different types have (in general) two different neighbourhood profiles. It is also the case after interleaving the parity-check equations of $\mathcal{C}_\pi$.
Therefore we can classify parity-check equations into groups using their neighbourhood profiles. Of course, parity-check equations involving extreme positions of $\boldsymbol{x}$ (the first or last) do not have exactly the same neighbourhood profile as the other parity-check equations of the same type. These parity-check equations have lost parity-check equations in their neighbourhood. This motivates to bring in the following partial order on the profile of parity-check equations

Definition 6 (Partial order on the profiles of parity-check equations)**.** *We define a partial order: $\mathcal{P} \leq \mathcal{P}'$ if $\forall i \leq t$, $\mathcal{P}_i \leq \mathcal{P}'_i$.*

Classifying a given parity-check equation. The algorithm for classifying parity-check equations into groups is given by Algorithm 1. With this algorithm we also deduce the length n of the convolutional code $\mathcal{C}$.

Remark 1 (Choose a group)**.** *We form r groups, but we just need one of them. We choose a group that minimizes the intersection number of its parity-check equations. This is a heuristic whose rationale is that the group with the smallest intersection number corresponds probably to the equivalence class with the smallest span. Indeed, in $\mathcal{C}$, the larger the span of a parity-check equation $\mathcal{E}$ is, the more chances we have that there are parity-check equations with at least one position in common with $\mathcal{E}$.*

Remark 2 (Deducing the length n of $\mathcal{C}$)**.** *The number nb_{eq} of parity-check equations in the group that we keep allows us to deduce the size n of the convolutional code $\mathcal{C}$, $n = \lfloor \frac{N}{nb_{eq}} \rfloor$. This equality is due to the fact that almost all parity-check equations in this group correspond (after deinterleaving) to shifts by a multiple of n of a single parity-check equation.*

4.3 Recovering a parity-check equation of the convolutional code

From now on, we assume that we have a set $\mathcal{L}_1$ of parity-check equations of $\mathcal{C}_\pi$. These parity-check equations are of weight t and in the same equivalence

Algorithm 1: Classifying parity-check equations and deducing n

input: $\mathcal{L}$ a set of parity-check equations of $\mathcal{C}_\pi$

output:

- $\mathcal{L}_1$ a set of parity-check equations of the same type
- the length n of the convolutional code.

for *all* $\mathcal{E} \in \mathcal{L}$ **do**
  $\mathcal{P}^{\mathcal{E}} \leftarrow$ the neigbourhood profiles of $\mathcal{E}$
$\mathcal{P}^{E_1}, \ldots, \mathcal{P}^{E_r} \leftarrow$ most frequent profiles in $\{\mathcal{P}^{\mathcal{E}}, \mathcal{E} \in \mathcal{L}\}$
$\mathcal{L}_1, \ldots, \mathcal{L}_r \leftarrow \emptyset$
for *all* $\mathcal{E} \in \mathcal{L}$ **do**
  if $\mathcal{P}^{E_i}$ *is the unique profile such that* $\mathcal{P}^{\mathcal{E}} \leq \mathcal{P}^{E_i}$ **then**
    $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{\mathcal{E}\}$
for *all* $i \in \{1, \ldots, r\}$ **do**
  $\mathcal{I}^{E_i} \leftarrow \sum_{j \leq t} \mathcal{P}_j^{E_i}$
$\mathcal{L}_1 \leftarrow \mathcal{L}_i$ with i such that $\mathcal{I}^{E_i} = \min_j \mathcal{I}^{E_j}$
$n \leftarrow \lfloor \frac{N}{\#\mathcal{L}_1} \rfloor$
return $\mathcal{L}_1$ and n

class: they correspond to parity-checks of $\mathcal{C}$ which are shifts of each other by a multiple of n.

We denote by $\mathcal{E}_{\mathcal{C}}$ a parity-check equation of $\mathcal{C}$ such that each parity-check equation $\mathcal{E}$ of $\mathcal{L}_1$ satisfies $\pi^{-1}(\mathcal{E}) \sim \mathcal{E}_{\mathcal{C}}$ (that is each parity-check equation $\mathcal{E}$ of $\mathcal{L}_1$ is such that $\pi^{-1}(\mathcal{E})$ is a shift of $\mathcal{E}_{\mathcal{C}}$).

The purpose of this subsection is to show how $\mathcal{E}_{\mathcal{C}}$ can be recovered from the knowledge of $\mathcal{L}_1$. $\mathcal{E}_{\mathcal{C}}$ is the parity-check equation of a sub-code of $\mathcal{C}$, this sub-code is an $(n, n-1)$ convolutional code. To recover this $(n, n-1)$ convolutional code we test each $(n, n-1)$ convolutional code that admits a parity-check equation of weight t and with a span less than $s_{\max}$ where $s_{\max}$ is some chosen constant. Our strategy is to attach a graph to a set of parity-check equations such that
(i) the equivalence class of a parity-check equation $\mathcal{E}$ of an $(n, n-1)$ convolutional code discriminates the convolutional code
(ii) if two sets of parity-check equations differ from each other by a permutation then their associated graphs are isomorphic.

By checking if there is an isomorphism between the graph associated to $\mathcal{L}_1$ and the graph associated to shifts of a parity-check equation of an $(n, n-1)$ convolutional code we will recover the right convolutional code and adding labels to the graph will allow us to identify the permutation between the two sets of parity-check equations.

Graphs associated to $\mathcal{L}_1$ and $\mathcal{E}$

From now on we will use the following notation

Notation We denote by ℓ the number of parity-check equations in $\mathcal{L}_1$.

To the set of parity-check equations $\mathcal{L}_1$ we associate a labeled graph $\tilde{\mathcal{G}}(\mathcal{L}_1)$ which is defined as follows

Definition 7 (Graph associated to a set of parity-check equations)**.** *The labeled graph $\tilde{\mathcal{G}}(\mathcal{L})$ associated to a set $\mathcal{L}$ of parity-check equations is such that*

- *Each parity-check equation of $\mathcal{L}$ is represented in $\tilde{\mathcal{G}}(\mathcal{L})$ by a vertex.*
- *If $\mathcal{E}$ and $\mathcal{E}'$, two parity-check equations of $\mathcal{L}$, have k positions in common (that is $|\mathcal{E} \cap \mathcal{E}'| = k$) then in $\tilde{\mathcal{G}}(\mathcal{L})$ the two corresponding vertices are connected by k edges.*

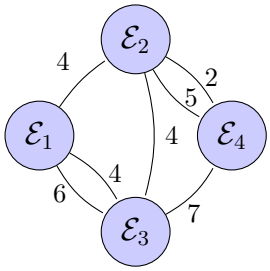

Figure 2: Graph $\tilde{\mathcal{G}}$ associated to $\mathcal{L} = \{\mathcal{E}_1 = \{1,4,6\}, \mathcal{E}_2 = \{2,4,5\}, \mathcal{E}_3 = \{4,6,7\}, \mathcal{E}_4 = \{2,5,7\}\}$

- *Each edge of $\tilde{\mathcal{G}}(\mathcal{L})$ is labeled with the number of the position that it represents.*

When $\mathcal{L}$ is clear from the context we will just denote this graph by $\tilde{\mathcal{G}}$.

Notation We denote by $\mathcal{G}$ the graph $\tilde{\mathcal{G}}$ without label on edges.

Example 2. *Let $\mathcal{L} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$ with $\mathcal{E}_1 = \{1,4,6\}$, $\mathcal{E}_2 = \{2,4,5\}$, $\mathcal{E}_3 = \{4,6,7\}$ and $\mathcal{E}_4 = \{2,5,7\}$. The graph $\tilde{\mathcal{G}}$ associated to $\mathcal{L}$ is represented on Figure 2.*

The graph $\tilde{\mathcal{G}}(\mathcal{L}_1)$ associated to $\mathcal{L}_1$ represents the interleaved sub-code of $\mathcal{C}$. To recover this sub-code (not interleaved) we test each $(n, n-1)$ convolutional code. This is achieved as follows. An $(n, n-1)$ convolutional code is defined by a parity-check equation $\mathcal{E} = \{e_1, \ldots, e_t\}$. Using $\mathcal{E}$ we construct a set $\mathcal{L}_\mathcal{E}$ of parity-check equations of this $(n, n-1)$ convolutional code: $\mathcal{L}_\mathcal{E} = \{\{e_1 + in, e_2 + in, \ldots, e_t + in\}, -\frac{\ell}{n} \leq i < \frac{\ell}{n}\}$. $\mathcal{L}_\mathcal{E}$ contains ℓ consecutive parity-check equations obtained by shifts of $\mathcal{E}$ by a multiple of n. Using Definition 7 we associate the graph $\tilde{\mathcal{G}}(\mathcal{L}_\mathcal{E})$ to this set $\mathcal{L}_\mathcal{E}$. To simplify notation we denote this graph by $\tilde{\mathcal{G}}^\mathcal{E}$.
We want to compare this graph $\tilde{\mathcal{G}}^\mathcal{E}$ to $\tilde{\mathcal{G}}(\mathcal{L}_1)$, it is for this reason that we take ℓ parity-check equations in $\mathcal{L}_\mathcal{E}$, so the two graphs have the same number of vertices, and we check if they are isomorphic.

Definition 8 (Isomorphic graphs)**.** *Two graphs $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{G}}'$ are isomorphic if, and only if, there exists a bijective mapping ϕ between the vertices of $\tilde{\mathcal{G}}$ and the vertices of $\tilde{\mathcal{G}}'$ so that for any pair of vertices (x, y) of $\tilde{\mathcal{G}}$ there is the same number of edges between x and y as there are edges between $\phi(x)$ and $\phi(y)$ in $\tilde{\mathcal{G}}'$.*

We we also need a finer definition of isomorphism which is suitable for labeled graphs

Definition 9 (Equivalent graphs)**.** *Two labeled graphs $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{G}}'$ are equivalent if they are isomorphic (call the corresponding mapping ϕ) and there exists a bijective mapping ψ of the labels from one graph to the other so that for any pair of vertices (x, y) of $\tilde{\mathcal{G}}$ if we denote by $\{a_1, \ldots, a_s\}$ the (multi)set of labels of the edges between x and y, then the edges between $\phi(x)$ and $\phi(y)$ in $\tilde{\mathcal{G}}'$ have labels $\{\psi(a_1), \ldots, \psi(a_s)\}$.*

To recover the parity-check equation $\mathcal{E}_{\mathcal{C}}$ of the sub-code of $\mathcal{C}$ and the interleaver π we use the following proposition

Proposition 10. *If $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ then $\tilde{\mathcal{G}}^{\mathcal{E}}$ and $\tilde{\mathcal{G}}(\mathcal{L}_1)$ are equivalent.*

Proof. We assume that $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$. Let $\mathcal{L}_{\mathcal{E}} = \{\mathcal{E}_0, \ldots, \mathcal{E}_{\ell-1}\}$ be the set of parity-check equations associated to $\mathcal{E}$, with $\mathcal{E}_i$ being equal to $\mathcal{E}_{i-1}$ shifted by n. $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ so $\mathcal{L}_1$ contains the same parity-check equations as $\mathcal{L}_{\mathcal{E}}$ but interleaved by the interleaver π: $\mathcal{L}_1 = \{\pi(\mathcal{E}_0), \pi(\mathcal{E}_1), \ldots, \pi(\mathcal{E}_{\ell-1})\}$. Note that the interleaver changes the numbering of positions, not the number of positions in common between two parity-check equations. An isomorphism ϕ between vertices of this graph is given by $\phi : \tilde{\mathcal{G}}^{\mathcal{E}} \to \tilde{\mathcal{G}}(\mathcal{L}_1)$, $\mathcal{E}_i \mapsto \pi(\mathcal{E}_i)$, $\forall i < \ell - 1$. This shows that that $\tilde{\mathcal{G}}^{\mathcal{E}}$ and $\tilde{\mathcal{G}}(\mathcal{L}_1)$ are isomorphic. If we denote by $m^{\mathcal{E}}$ the minimal value such that $\tilde{\mathcal{G}}^{\mathcal{E}}$ contains a vertex representing a parity-check equation involving the position $m^{\mathcal{E}}$, the block interleaver π gives an isomorphism ψ on labels between $\tilde{\mathcal{G}}^{\mathcal{E}}$ and $\tilde{\mathcal{G}}^{\pi}$. $\psi : \tilde{\mathcal{G}}^{\mathcal{E}} \to \tilde{\mathcal{G}}(\mathcal{L}_1)$, $i \mapsto \pi(i - m^{\mathcal{E}})$. We obtain that $\tilde{\mathcal{G}}^{\mathcal{E}}$ and $\tilde{\mathcal{G}}(\mathcal{L}_1)$ are equivalent. □

Remark 3. *If we find a parity-check equation $\mathcal{E}$ such that $\tilde{\mathcal{G}}(\mathcal{L}_1)$ and $\tilde{\mathcal{G}}^{\mathcal{E}}$ are equivalent, then an isomorphism ψ between labels of these graphs gives the block interleaver Π such that $\Pi(\mathcal{C}) = \mathcal{C}_{\pi}$.*

Sub-graphs associated to $\mathcal{L}_1$ and $\mathcal{E}$

To check the equivalence we will need auxiliary graphs which are much smaller and that will in general be sufficient for testing the equivalence between graphs. More precisely, we use sub-graphs induced by $\tilde{\mathcal{G}}^{\mathcal{E}}$ and $\tilde{\mathcal{G}}(\mathcal{L}_1)$.

Notation From now on to simplify notation we will denote the graph $\tilde{\mathcal{G}}(\mathcal{L}_1)$ by $\tilde{\mathcal{G}}^{\pi}$.

We will associate six sub-graphs, $\mathcal{G}_1^{\pi}, \mathcal{G}_2^{\pi}, \tilde{\mathcal{G}}_2^{\pi}, \mathcal{G}_1^{\mathcal{E}}, \mathcal{G}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\mathcal{E}}$, to $\mathcal{L}_1$ and $\mathcal{E}$ such that if $\tilde{\mathcal{G}}^{\mathcal{E}}$ and $\tilde{\mathcal{G}}^{\pi}$ are equivalent then:

- $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\pi}$ are isomorphic
- $\mathcal{G}_2^{\mathcal{E}}$ and $\mathcal{G}_2^{\pi}$ are isomorphic
- $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\pi}$ are equivalent

The first graphs $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\pi}$ are not labeled and represent the neighbourhood of a parity-check equation.

To obtain $\mathcal{G}_1^{\pi}$, we randomly choose a parity-check equation $\mathcal{E}_0$ in $\mathcal{L}_1$. $\mathcal{G}_1^{\pi}$ is the sub-graph of $\mathcal{G}^{\pi}$ induced by the vertex representing $\mathcal{E}_0$ and all vertices having at least one edge in common with it.
$\mathcal{G}_1^{\mathcal{E}}$ is a sub-graph of $\mathcal{G}^{\mathcal{E}}$ induced by vertices representing $\mathcal{E}$ and all its shifts by a multiple of n such that they have at least one position in common with $\mathcal{E}$. This graph contains only a small number of vertices as shown by

Proposition 11. *Let $\mathcal{E}$ be a parity-check equation of an $(n, n-1)$ convolutional code and $s^{\mathcal{E}}$ the span of $\mathcal{E}$. The sub-graph $\mathcal{G}_1^{\mathcal{E}}$ associated to $\mathcal{E}$ contains at most $2\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 1$ vertices. (In other words, the parity-check equation $\mathcal{E}$ has at most $2\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 1$ parity-check equations in its neighbourhood.)*

Notation We denote by $\mathcal{E}^{(i)}$ the parity-check equation equals to $\mathcal{E}$ shifted by in.

Proof. Assume that the parity-check equation $\mathcal{E}$ is given by $\mathcal{E} = \{e_1, \ldots, e_t\}$ with $e_1 < e_2 < \cdots < e_t$. $s^{\mathcal{E}}$ is the span of $\mathcal{E}$, so $s^{\mathcal{E}} = e_t - e_1 + 1$. $\mathcal{E}^{(i)}$ is represented in $\mathcal{G}_1^{\mathcal{E}}$ if and only if $\mathcal{E}^{(i)}$ and $\mathcal{E}$ have at least one position in common, that is $\{e_1, \ldots, e_t\} \cap \{e_1 + in, \ldots, e_t + in\} \neq \emptyset$.
For $i \geq 0$ we have $\{e_1, \ldots, e_t\} \cap \{e_1 + in, \ldots, e_t + in\} \neq \emptyset$ if $e_t \geq e_1 + in$, that is $0 \leq i < \frac{s^{\mathcal{E}}}{n}$. So for $i \geq 0$, there are at most $\lceil \frac{s^{\mathcal{E}}}{n} \rceil$ parity-check equations which can have positions in commons with $\mathcal{E}$.
If $i < 0$. $\{e_1, \ldots, e_t\} \cap \{e_1 + in, \ldots, e_t + in\} \neq \emptyset$ if $e_t + in \geq e_1$, that is $-\frac{s^{\mathcal{E}}}{n} < i < 0$. For $i < 0$, there is at most $\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 1$ parity-check equations which can have positions in commons with $\mathcal{E}$.

So, the maximal number of parity-check equations which can have a position in common with $\mathcal{E}$ is equal to $2\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 1$. □

Proposition 12. *If $\pi^{-1}(\mathcal{E}_0)$ does not involve the first or last positions, and if $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ then $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\pi}$ are isomorphic.*

Proof. $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ (that is $\pi^{-1}(\mathcal{E}_0) \sim \mathcal{E}$), we denote by $\mathcal{E}'$ the parity-check equation $\mathcal{E}^{(k)}$ such that $\pi^{-1}(\mathcal{E}_0) = \mathcal{E}^{(k)}$. If $\pi^{-1}(\mathcal{E}_0)$ does not involve the first or last positions and if we denote by $I = \{i_1, \ldots, i_j\}$ the set of integers such as $\forall i \in I$, $\mathcal{E}'$ and $\mathcal{E}'^{(i)}$ have at least one position in common, then $\mathcal{L}_1$ contains $\pi(\mathcal{E}'^{(i_1)}), \ldots, \pi(\mathcal{E}'^{(i_j)})$. All these parity-check equations have at least one position in common with $\mathcal{E}_0$, so they are represented in $\mathcal{G}_1^{\pi}$ and we have an isomorphism ϕ between $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\pi}$ defined by $\phi : \mathcal{G}_1^{\mathcal{E}} \to \mathcal{G}_1^{\pi}$, $\mathcal{E}^{(i)} \mapsto \pi(\mathcal{E}^{(i)})$. □

So the first step to test a given $(n, n-1)$ convolutional code, consists in checking if $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\pi}$ are isomorphic. If $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\pi}$ are not isomorphic then $\mathcal{E} \neq \mathcal{E}_{\mathcal{C}}$. But these graphs are not enough discriminating, two $(n, n-1)$ convolutional codes, defined by $\mathcal{E}$ and $\mathcal{E}'$ can be associated to two isomorphic graphs $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\mathcal{E}'}$.

Example 3. *For $n = 2$, the graph $\mathcal{G}_1^{\mathcal{E}}$ associated to the parity-check equation $\mathcal{E} = \{1, 2, 4, 6, 7\}$ is isomorphic to the graph $\mathcal{G}_1^{\mathcal{E}'}$ associated to the parity-check equation $\mathcal{E}' = \{1, 3, 4, 6, 7\}$. These graphs are represented on Figure 3. An isomorphism between these graphs is defined by $\phi : \mathcal{G}_1^{\mathcal{E}} \to \mathcal{G}_1^{\mathcal{E}'}$, $\mathcal{E}^{(i)} \mapsto \mathcal{E}'^{(i)}$.*

We associate to $\mathcal{L}_1$ and $\mathcal{E}$ two other graphs $\mathcal{G}_2^{\pi}$ and $\mathcal{G}_2^{\mathcal{E}}$. These graphs are not labeled and represent the neighbourhood at distance two of a parity-check equation.

$\mathcal{G}_2^{\pi}$ is the sub-graph of $\mathcal{G}^{\pi}$ induced by $\mathcal{G}_1^{\pi}$ and all vertices having at least one edge in common with a vertex of $\mathcal{G}_1^{\pi}$. So $\mathcal{G}_2^{\pi}$ represents the neighbourhood at distance 2 of $\mathcal{E}_0$ in $\mathcal{L}_1$.
$\mathcal{G}_2^{\mathcal{E}}$ is the sub-graph of $\mathcal{G}^{\mathcal{E}}$ induced by $\mathcal{G}_1^{\mathcal{E}}$ and all vertices having at least one edge in common with a vertex of $\mathcal{G}_1^{\pi}$. This graph is rather small too as shown by:

Proposition 13. *Let $\mathcal{E}$ be a parity-check equation of an $(n, n-1)$ convolutional code and $s^{\mathcal{E}}$ be the span of $\mathcal{E}$. The sub-graph $\mathcal{G}_2^{\mathcal{E}}$ associated to $\mathcal{E}$*

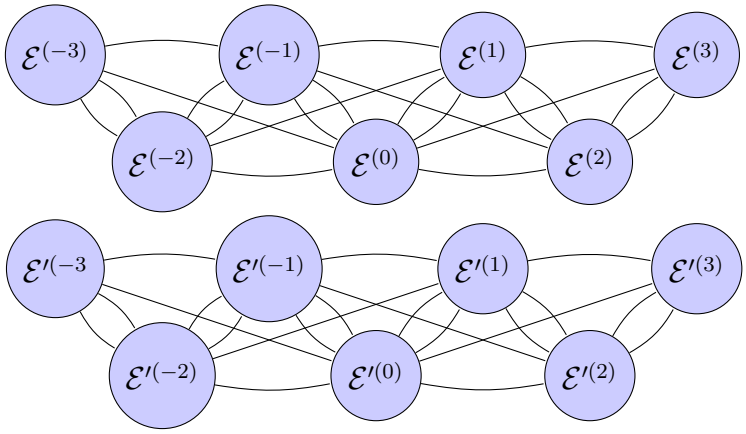

Figure 3: Graphs $\mathcal{G}_1^{\mathcal{E}}$ and $\mathcal{G}_1^{\mathcal{E}'}$ with $n = 2$, $\mathcal{E} = \{1, 2, 4, 6, 7\}$ and $\mathcal{E}' = \{1, 3, 4, 6, 7\}$.

contains at most $4\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 3$ *vertices. (In other words, the parity-check equation* $\mathcal{E}$ *has at most* $4\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 3$ *parity-check equations in its neighbourhood at distance 2.)*

Proposition 14. *If* $\pi^{-1}(\mathcal{E}_0)$ *does not involve the first or last positions, and if* $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ *then* $\mathcal{G}_2^{\mathcal{E}}$ *and* $\mathcal{G}_2^{\pi}$ *are isomorphic.*

Proof. $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ (that is $\pi^{-1}(\mathcal{E}_0) \sim \mathcal{E}$), we denote by $\mathcal{E}'$ the parity-check equation $\mathcal{E}^{(k)}$ such that $\pi^{-1}(\mathcal{E}_0) = \mathcal{E}^{(k)}$. If $\pi^{-1}(\mathcal{E}_0)$ does not involve the first or last positions and if we denote by $I = \{i_1, \ldots, i_j\}$ the set of integers such that for all i in I, $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ contains a vertex representing $\mathcal{E}^{(i)}$, then $\mathcal{L}_1$ contains $\pi(\mathcal{E}'^{(i_1)}), \ldots, \pi(\mathcal{E}'^{(i_j)})$. All these parity-check equations are in the neighbourhood at distance 2 of $\mathcal{E}_0$, so they are represented in $\mathcal{G}_2^{\pi}$ and we have an isomorphism ϕ between $\mathcal{G}_2^{\mathcal{E}}$ and $\mathcal{G}_2^{\pi}$ defined by $\phi : \mathcal{G}_2^{\mathcal{E}} \to \mathcal{G}_2^{\pi}$, $\mathcal{E}^{(i)} \mapsto \pi(\mathcal{E}^{(i)})$. □

The second step to test a given $(n, n-1)$ convolutional code, consists in checking if $\mathcal{G}_2^{\mathcal{E}}$ and $\mathcal{G}_2^{\pi}$ are isomorphic but these graphs are not sufficiently discriminating. Finally we use two small labeled graphs $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\pi}$.

To obtain $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\pi}$ we just add label on edges of $\mathcal{G}_2^{\mathcal{E}}$ and $\mathcal{G}_2^{\pi}$.

Proposition 15. *If* $\pi^{-1}(\mathcal{E}_0)$ *does not involve the first or last positions, and if* $\mathcal{E} = \mathcal{E}_{\mathcal{C}}$ *then* $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ *and* $\tilde{\mathcal{G}}_2^{\pi}$ *are equivalent.*

Proof. With Proposition 14 we deduce that $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\pi}$ are equivalent. If we denote by $m^{\mathcal{E}}$ and m^{π} the minimal values such that $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\pi}$ contain a vertex representing a parity-check equation involving respectively positions $m^{\mathcal{E}}$ and m^{π}, then we have an isomorphism ψ between labels of $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\pi}$: $\psi : \tilde{\mathcal{G}}_2^{\mathcal{E}} \to \tilde{\mathcal{G}}_2^{\pi}$, $i \mapsto \pi(i - m^{\mathcal{E}} + \pi^{-1}(m^{\pi}))$. □

Finally the algorithm used for recovering the parity-check equation $\mathcal{E}_{\mathcal{C}}$ of the sub-code of $\mathcal{C}$ is Algorithm 2.

Algorithm 2: Recovering the parity-check equation $\mathcal{E}_{\mathcal{C}}$

input: $\mathcal{L}_1$ a set of parity-check equations of weight t of the same type and n the length of $\mathcal{C}$
output: L a list of parity-check equations such that $\mathcal{E}_{\mathcal{C}}$ can be equal to each of them
$L \leftarrow \emptyset$
$\mathcal{E}_0 \leftarrow$ choose at random a parity-check equation of $\mathcal{L}_1$
$\mathcal{G}_1^{\pi}, \mathcal{G}_2^{\pi}$ and $\tilde{\mathcal{G}}_2^{\pi} \leftarrow$ sub-graphs induced by $\mathcal{E}_0$ and its neighbourhood
for *all* $\mathcal{E}$ *of weight* t *and with a span less than* s_{max} **do**
  $\mathcal{G}_1^{\mathcal{E}} \leftarrow$ graph representing the neighbourhood of $\mathcal{E}$ at distance 1
  if $\mathcal{G}_1^{\mathcal{E}}$ *and* $\mathcal{G}_1^{\pi}$ *are isomorphic* **then**
    $\mathcal{G}_2^{\mathcal{E}} \leftarrow$ graph representing the neighbourhood at distance 2 of $\mathcal{E}$
    if $\mathcal{G}_2^{\mathcal{E}}$ *and* $\mathcal{G}_2^{\pi}$ *are isomorphic* **then**
      $\tilde{\mathcal{G}}_2^{\mathcal{E}} \leftarrow$ labeled graph representing the neighbourhood at distance 2 of $\mathcal{E}$
      if $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ *and* $\tilde{\mathcal{G}}_3^{\pi}$ *are equivalent* **then**
        $L \leftarrow L \cup \{\mathcal{E}\}$
return L

Reducing the number of tests

In fact, these graphs have lots of symmetries, and we do not really need to test all parity-check equations of weight t and with a span less than s_{max}. The following proposition allows us to reduce the number of $(n, n-1)$ convolutional code that we have to test.

Proposition 16. *Let* $\mathcal{E} = \{e_1, \ldots, e_t\}$ *be the parity-check equation of an* $(n, n-1)$ *convolutional code and* $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ *be the labelled graph representing the*

neighbourhood at distance two of $\mathcal{E}$.
$\mathcal{E}$ *can also be represented by a binary vector* $b_1 \dots b_s$ *where* $b_i = 1$ *if* $i \in \{e_1, \dots, e_t\}$.

- *The graph* $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ *associated to* $\mathcal{E}'$ *represented by the binary vector* $b_s \dots b_1$ *is equivalent to* $\tilde{\mathcal{G}}_2^{\mathcal{E}}$.
- *For all permutations* $p = [p_1, \dots, p_n]$ *of length* n, *the graph* $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ *associated to the parity-check equation* $\mathcal{E}'$ *represented by the binary vector* $p(b_1 \dots b_n)p(b_{n+1} \dots b_{2n}) \dots p(b_{s-n+1} \dots b_s)$ *is equivalent to* $\tilde{\mathcal{G}}_2^{\mathcal{E}}$.

Proof.
- For the first point, if $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ contains $\mathcal{E}^{(i_1)}, \dots, \mathcal{E}^{(i_j)}$ then $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ contains $\mathcal{E}'^{(i_1)}, \dots, \mathcal{E}'^{(i_j)}$, and between these two graphs we have the isomorphism ϕ defined by $\phi : \tilde{\mathcal{G}}_2^{\mathcal{E}} \to \tilde{\mathcal{G}}_2^{\mathcal{E}'}$, $\mathcal{E}^{(i)} \mapsto \mathcal{E}'^{(i_j - i)}$ for all $i \in \{i_1, \dots, i_j\}$. The isomorphism ψ between labels of these graphs can be defined by $\psi : \tilde{\mathcal{G}}_2^{\mathcal{E}} \to \tilde{\mathcal{G}}_2^{\mathcal{E}'}$, $k \mapsto m^{\mathcal{E}} - k$ where $m^{\mathcal{E}}$ is the maximal value of labels of $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$. With these two isomorphisms we deduce that $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ are equivalent.
- For the second point, if $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ contains $\mathcal{E}^{(i_1)}, \dots, \mathcal{E}^{(i_j)}$ then $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ contains $\mathcal{E}'^{(i_1)}, \dots, \mathcal{E}'^{(i_j)}$, and between these two graphs we have the isomorphism ϕ defined by $\phi : \tilde{\mathcal{G}}_2^{\mathcal{E}} \to \tilde{\mathcal{G}}_2^{\mathcal{E}'}$, $\mathcal{E}^{(i)} \mapsto \mathcal{E}'^{(i)}$ for all $i \in \{i_1, \dots, i_j\}$. We define the permutation P by $P(i) = p(i \mod n) + \lfloor \frac{i}{n} \rfloor$. The isomorphism ψ on labels defined by $\psi : \tilde{\mathcal{G}}_2^{\mathcal{E}} \to \tilde{\mathcal{G}}_2^{\mathcal{E}'}$, $k \mapsto P(k)$ allows us to deduce that $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ and $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ are equivalent.

□

Definition 17 (Equivalent parity-check equations)**.** *Let* $\mathcal{E}$ *and* $\mathcal{E}'$ *be two parity-check equations, if using the Proposition 16 we can deduce that the two graphs* $\tilde{\mathcal{G}}_2^{\mathcal{E}}$ *and* $\tilde{\mathcal{G}}_2^{\mathcal{E}'}$ *are equivalent we say that* $\mathcal{E}$ *and* $\mathcal{E}'$ *are equivalent.*

Example 4. *For* $n = 2$, *if* $s_{max} = 20$ *and* $t = 10$ *we run only* 15 328 *tests instead of* 184 756. *If we suppose that* $s_{max} = 30$, 1 238 380 *tests are needed instead of* 30 045 015.
If the sought parity-check equation is $\mathcal{E}_{\mathcal{C}} = \{1, 2, 3, 5, 6, 7, 8, 12, 13, 14\}$ *(of weight 10), only 2 parity-check equations produce an isomorphic graph to* $\mathcal{G}_1^{\pi}$ *and among them one is equivalent to* $\tilde{\mathcal{G}}_2^{\pi}$ *for* $s_{max} = 20$ *(testing the* 15 328 *parity-check equations takes approximately 1 second). If we take* $s_{max} = 30$, *4 graphs are isomorphic to* $\mathcal{G}_1^{\pi}$ *and 2 are equivalent to* $\tilde{\mathcal{G}}_2^{\pi}$, *(one of them is eliminated later) these tests take less than 3 minutes.*

If the parity-check equation $\mathcal{E}_\mathcal{C}$ is not recovered

If no parity-check equation has an equivalent labeled graph with $\tilde{\mathcal{G}}_2^\pi$ we may have chosen in $\mathcal{L}_1$ a parity-check equation $\mathcal{E}_0$ which has incomplete graphs (after deinterleaving this parity-check equation involves the first or last positions of $\boldsymbol{x}$ or at least a parity-check equation in its neighbourhood at distance 2 is missing in $\mathcal{L}_1$).

In this case, we randomly choose another parity-check equation in $\mathcal{L}_1$, we compute the new graphs $\mathcal{G}_1^\pi$, $\mathcal{G}_2^\pi$ and $\tilde{\mathcal{G}}_2^\pi$ representing its neighbourhood at distance one and two, and we test all convolutional codes.

Remark 4. *If $\mathcal{G}_1^\pi$ or $\mathcal{G}_2^\pi$ is incomplete, it is probably not symmetric so, no graph $\mathcal{G}_1^\mathcal{E}$ or $\mathcal{G}_2^\mathcal{E}$ can be isomorphic with it and the test of all $(n, n-1)$ convolutional codes is very quickly (we just compare the number of vertices, they have not the same number so they can't be isomorphic).*

If $\mathcal{E}_\mathcal{C}$ can be equal to several parity-check equations $\mathcal{E}$ we apply the end of the method for each of them.

4.4 Ordering parity-check equations

Using $\mathcal{E}_\mathcal{C}$ the parity-check equation previously recovered, we want to order the parity-check equations of $\mathcal{L}_1$. That is, find an ordering $\mathcal{A} = \mathcal{E}_{a_1}, \ldots, \mathcal{E}_{a_l}$ of these parity-check equations such that $\pi^{-1}(\mathcal{E}_{a_{i+1}})$ is equal to the shift by n of $\pi^{-1}(\mathcal{E}_{a_i})$. All parity-check equations of $\mathcal{L}_1$ belong to $\mathcal{A}$ once and only once.

To order these parity-check equations we extend the two graphs $\mathcal{G}_2^\pi$ and $\mathcal{G}_2^{\mathcal{E}_\mathcal{C}}$ and we search for an isomorphism between the vertices of these two extended graphs. This isomorphism give us the ordering $\mathcal{A}$.

When we recover the parity-check equation $\mathcal{E}_\mathcal{C}$ of the sub-code of $\mathcal{C}$ we search for an isomorphism between $\mathcal{G}_2^\pi$ and $\mathcal{G}_2^{\mathcal{E}_\mathcal{C}}$. Once we know this isomorphism, we also have the bijection ϕ between the vertices of these graphs. This bijection gives us a part of the ordering. Indeed, for all i, such that $\mathcal{G}_2^{\mathcal{E}_\mathcal{C}}$ contains a vertex V_i representing $\mathcal{E}_\mathcal{C}$ shifted by in, we place the parity-check

equation represented by $\phi(V_i)$ at position i in $\mathcal{A}$.

To obtain the bijection using all parity-check equations of $\mathcal{L}_1$ and deduce the entire ordering $\mathcal{A}$, we extend step by step $\mathcal{G}_2^{\pi}$, $\mathcal{G}_2^{\mathcal{E}_\mathcal{C}}$ and the bijection ϕ. We denote by $\mathcal{G}_{a..b}^{\mathcal{E}_\mathcal{C}}$ the graph representing $\mathcal{E}$ shifted by in for all integers $i \in [a, b]$, $\phi_{a..b}$ the isomorphism defined for all integers between a and b, and $\mathcal{G}_{a..b}^{\pi}$ the graph $\phi(\mathcal{G}_{a..b}^{\mathcal{E}_\mathcal{C}})$.

A step of the extension. Knowing $\mathcal{G}_{a..b}^{\mathcal{E}_\mathcal{C}}$, $\mathcal{G}_{a..b}^{\pi}$ and $\phi_{a..b}$, we search for $\mathcal{G}_{a..b+1}^{\mathcal{E}_\mathcal{C}}$, $\mathcal{G}_{a..b+1}^{\pi}$ and $\phi_{a..b+1}$.

- To obtain $\mathcal{G}_{a..b+1}^{\mathcal{E}_\mathcal{C}}$ from $\mathcal{G}_{a..b}^{\mathcal{E}_\mathcal{C}}$ we just add a vertex representing $\mathcal{E}_\mathcal{C}$ shifted by $(b+1)n$ and the corresponding edges.
- We search in $\mathcal{L}_1$ for a parity-check equation $\mathcal{E}_i$ which is not represented in $\mathcal{G}_{a..b}^{\pi}$ and such that if we add a vertex representing this parity-check equation and the corresponding edges to $\mathcal{G}_{a..b}^{\pi}$, then $\phi_{a..b+1}$ defined by $\phi_{a..b+1}(j) = \psi_{a..b}(j)$ for $j \in [a, \ldots, b]$ and $\phi_{a..b+1}(b+1) = i$ is an isomorphism between $\mathcal{G}_{a..b+1}^{\mathcal{E}_\mathcal{C}}$ and $\mathcal{G}_{a..b}^{\pi}$ extended with $\mathcal{E}_i$.

When we can not extend $\mathcal{G}_{a..b}^{\mathcal{E}_\mathcal{C}}$, $\mathcal{G}_{a..b}^{\pi}$ and $\phi_{a..b}$ such that $\mathcal{G}_{a..b+1}^{\mathcal{E}_\mathcal{C}}$ and $\mathcal{G}_{a..b+1}^{\pi}$ are isomorphic, we extend these graphs and the isomorphism in the other direction. In other words, we search for $\mathcal{G}_{a-1..b}^{\mathcal{E}_\mathcal{C}}$, $\mathcal{G}_{a-1..b}^{\pi}$ and $\phi_{a-1..b}$ from $\mathcal{G}_{a..b}^{\mathcal{E}_\mathcal{C}}$, $\mathcal{G}_{a..b}^{\pi}$ and $\phi_{a..b}$.

Remark 5 (Several parity-check equations)**.** *If at a given step, several parity-check equations $\mathcal{E}_i$ of $\mathcal{L}_1$ can be chosen, then we extend the two graphs and the isomorphism with feedback and finally we choose the biggest isomorphism and corresponding graphs.*

Remark 6 (No parity-check equation)**.** *If no parity-check equation in $\mathcal{L}_1$ satisfies all conditions, it might be that the sought parity-check equation is not in $\mathcal{L}_1$. This parity-check equation was not found using [6], or not classified in this group (it is an unclassified parity-check equation). In this case, we add a "missing parity-check equation" to $\mathcal{G}_{a..b}^{\pi}$, that is we add a vertex and edges to respect the regularity of $\mathcal{G}_{a..b}^{\pi}$ and we define $\phi_{a..b+1}(b+1) =$ "missing" or $\phi_{a-1..b}(a-1) =$ "missing". Then we continue the extension of graphs and ϕ.*

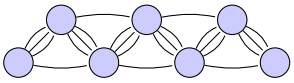

Figure 4: The starting graph $\mathcal{G}^{\mathcal{E}}_{a..b}$

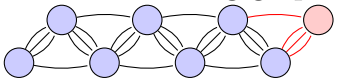

Figure 5: $\mathcal{G}^{\mathcal{E}}_{a..b+1}$ contains a missing parity-check equation

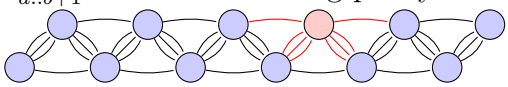

Figure 6: Continue to extend the graph

Remark 7. *We do not add more than* $\lceil \frac{s^{\mathcal{E}}}{n} \rceil - 1$ *consecutive "missing" parity-check equations in* $\mathcal{A}$ *because in this case, the next parity-check equation has no position in common with previous parity-check equations.*

Example 5. *We represent on Figures 4, 5 and 6 the extension with a missing parity-check equation (in red). There is an edge connecting a vertex lying before the missing parity-check equation to a vertex lying after this missing parity-check equation.*

At the end we recover the isomorphism between $\mathcal{G}^{\mathcal{E}}$ and $\mathcal{G}^{\pi}$. Indeed, at the end of the extension, $\mathcal{G}^{\mathcal{E}}_{a..b}$ is equal to $\mathcal{G}^{\mathcal{E}}$ and $\mathcal{G}^{\pi}_{a..b}$ to $\mathcal{G}^{\pi}$ probably with additional vertices representing missing parity-check equations.

4.5 Reconstructing the interleaver

Now we have the isomorphism between $\mathcal{G}^{\mathcal{E}_C}$ and $\mathcal{G}^{\pi}$, so to reconstruct the interleaver we need to recover the isomorphism ψ between labels on edges of $\tilde{\mathcal{G}}^{\mathcal{E}_C}$ and $\tilde{\mathcal{G}}^{\pi}$.

We recover ψ step by step, at each step we search for a sub-graph of $\tilde{\mathcal{G}}^{\mathcal{E}_C}$ which has a label i appearing only once, or appearing a different number of times than the other labels. The label of the image by ϕ of this edge labeled i gives us $\psi(i)$. Then we remove all edges labeled by i in $\mathcal{G}^{\mathcal{E}_C}$ and by $\psi(i)$ in $\tilde{\mathcal{G}}^{\mathcal{E}_\pi}$.

At the end of this extension, we extend ψ with positions which do not appear on graphs but are involved in parity-check equations represented by these graphs.

ψ defines the interleaver π, indeed $\pi(i) = \psi(i + m^{\mathcal{E}_C})$ where $m^{\mathcal{E}_C}$ is the minimal value such that $\mathcal{G}^{\mathcal{E}_C}$ contains a vertex representing a parity-check equation involving the position $m^{\mathcal{E}_C}$.

Remark 8 (Several isomorphisms)**.** *Depending on $\mathcal{E}_C$ there might be several bijections between labels of the two graphs. In this case we have several interleavers. For these interleavers only the first and last positions are different. The number of interleavers just depends on $\mathcal{E}_C$ and not on the length of π.*

Example 6. *The size of the interleaver is $N = 26$. The two graphs $\tilde{\mathcal{G}}^{\mathcal{E}_C}$ and $\tilde{\mathcal{G}}^{\pi}$ are represented on Figures 7 and 8. ϕ is defined by $\phi : \tilde{\mathcal{G}}^{\mathcal{E}_C} \to \tilde{\mathcal{G}}^{\pi}$, $\mathcal{E}_C^{(-4)} \mapsto \mathcal{E}_5$, $\mathcal{E}_C^{(-3)} \mapsto \mathcal{E}_6$, $\mathcal{E}_C^{(-2)} \mapsto \mathcal{E}_3$, ..., $\mathcal{E}_C^{(6)} \mapsto \mathcal{E}_2$.*
If we take the sub-graph of $\tilde{\mathcal{G}}^{\mathcal{E}_C}$ induced by $\mathcal{E}_C^{(-1)}$, $\mathcal{E}_C^{(0)}$ and $\mathcal{E}_C^{(1)}$, then we deduce that $\psi(3) = 1$ because the label 3 appears tree times and no other label appears tree times in this sub-graph. Then we also deduce that $\psi(1) = 3$, $\psi(4) = 25$ and $\psi(5) = 17$. With other sub-graphs we obtain the isomorphism ψ defined by $\psi : \tilde{\mathcal{G}}^{\mathcal{E}_C} \to \tilde{\mathcal{G}}^{\pi}$:

$-5 \mapsto 26$,	$0 \mapsto 20$,	$4 \mapsto 25$,	$8 \mapsto 13$,	$12 \mapsto 19$,
$-3 \mapsto 12$,	$1 \mapsto 3$,	$5 \mapsto 17$,	$9 \mapsto 11$,	$13 \mapsto 2$,
$-2 \mapsto 8$,	$2 \mapsto 15$,	$6 \mapsto 23$,	$10 \mapsto 7$,	$14 \mapsto 21$,
$-1 \mapsto 5$,	$3 \mapsto 1$,	$7 \mapsto 6$,	$11 \mapsto 16$,	$15 \mapsto 10$.

With this bijection we deduce a part of the interleaver π :
$\pi = [..., 26, ?, 12, 8, 5, 20, 3, 15, 1, 25, 17, 23, 6, 13, 11, 7, 16, 19, 2, 21, 10, ...]$
On the graphs we do not represent the positions involved in a single parity-check equation, we do not have edges with the corresponding label. But we know these values and we use them to determine the first and last positions of π. For example, the second parity-check equation represented in $\tilde{\mathcal{G}}^{\mathcal{E}_C}$ is $\mathcal{E}_C^{(-3)} = \{-5, -4, -3, -1, 0\}$ and the image by ϕ of this parity-check equation is $\mathcal{E}_6 = \{5, 12, 18, 20, 26\}$. With this parity-check equation, we extend the bijection with $-4 \mapsto 18$ (the only two unused values in these parity-check equations). With the same reasoning we deduce that $16 \mapsto 4$. The first parity-check equation represented on $\tilde{\mathcal{G}}^{\mathcal{E}_C}$ is $\mathcal{E}_C^{(-4)} = \{-7, -6, -5, -3, -2\}$ and the

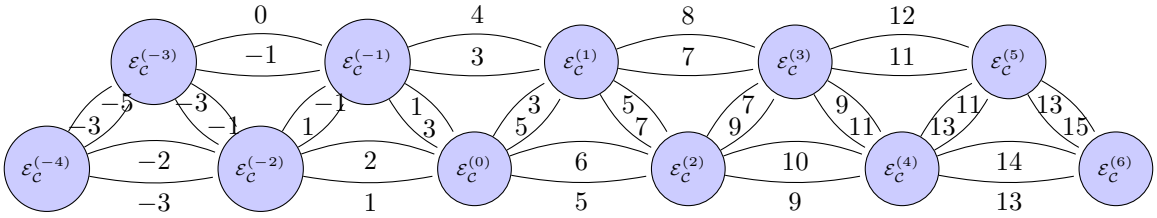

Figure 7: The graph $\tilde{\mathcal{G}}_2^{\mathcal{E}_C}$

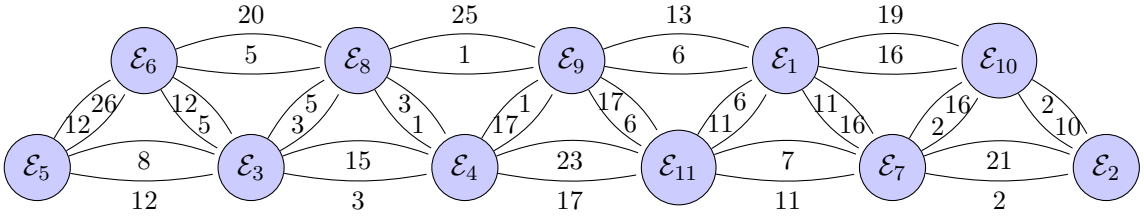

Figure 8: The graph $\tilde{\mathcal{G}}_2^{\mathcal{E}}$

corresponding equation on $\tilde{\mathcal{G}}^\pi$ *is* $\mathcal{E}_5 = \{8, 9, 12, 14, 26\}$*, with these parity-check equations we deduce that* $-7 \mapsto 9$ *and* $-6 \mapsto 14$ *or* $-7 \mapsto 14$ *and* $-6 \mapsto 9$*, we have the same indeterminate for the two last positions. So we have 4 possible interleavers:*
$\pi = [14, 9, 26, 18, 12, 8, 5, 20, 3, 15, 1, 25, 17, 23, 6, 13, 11, 7, 16, 19, 2, 21, 10, 4, 22, 24]$
or $\pi = [9, 14, 26, 18, 12, 8, 5, 20, 3, 15, 1, 25, 17, 23, 6, 13, 11, 7, 16, 19, 2, 21, 10, 4, 22, 24]$
or $\pi = [14, 9, 26, 18, 12, 8, 5, 20, 3, 15, 1, 25, 17, 23, 6, 13, 11, 7, 16, 19, 2, 21, 10, 4, 24, 22]$
or $\pi = [9, 14, 26, 18, 12, 8, 5, 20, 3, 15, 1, 25, 17, 23, 6, 13, 11, 7, 16, 19, 2, 21, 10, 4, 24, 22]$
Moreover we can also take the mirror of the isomorphism ϕ*, and we obtain 4 new possible interleavers.*

4.6 Particular cases

Indeterminate positions. If the reconstructed interleaver contains indeterminate positions, we search for these positions using noisy interleaved codewords. At each indeterminate positions we test all possible values. To test a position we reconstruct the missing parity-check equations and we verify the number of noisy interleaved codewords that satisfy these parity-check equations. If this number is less than a threshold (we can take $\frac{1}{2}\frac{1-(1-2\tau)^t}{2}M$)

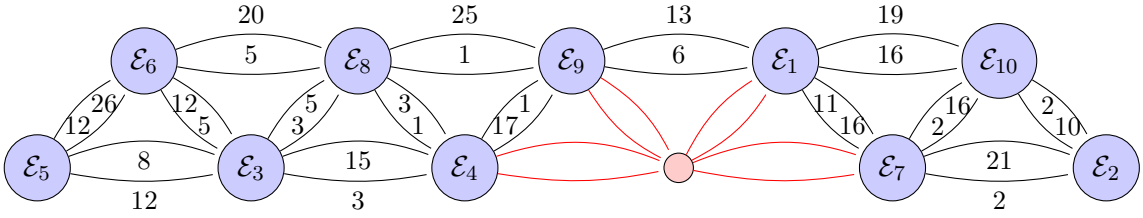

Figure 9: The graph $\tilde{\mathcal{G}}_2^{\mathcal{E}}$

it is not the correct value for this position.

Example 7. *The graph associated to $\tilde{\mathcal{G}}_2^{\mathcal{E}c}$ is on Figure 7, and the graph $\tilde{\mathcal{G}}_2^{\pi}$ on Figure 9. One bijection on edges ψ is the same as in Example 6, but we can not know if $7 \mapsto 6$ and $8 \mapsto 13$ or $7 \mapsto 13$ and $8 \mapsto 6$. To determine the right bijection we reconstruct the missing parity-check equation and we test them with noisy interleaved codewords. We test $7 \mapsto 6$ and $8 \mapsto 13$, in this case the missing equation is $\{6, 7, 11, 17, 23\}$, then we test $7 \mapsto 13$ and $8 \mapsto 6$, the missing equation is $\{7, 11, 13, 17, 23\}$. With the number of noisy interleaved codewords that satisfy these equations we deduce the 8 possible interleavers as in Example 6.*

Not the right length. If the reconstructed interleaver has not the right length, it is the case when $\mathcal{L}_1$ does not contain all parity-check equations of the same type, the missing parity-check equations are not classified. To recover the beginning and the end of the interleaver we continue the reconstruction by applying the same steps using unclassified parity-check equations: we extend the graphs $\mathcal{G}_{a..b}^{\pi}$ and $\mathcal{G}_{a..b}^{\mathcal{E}c}$ then we label these graphs and deduce the entire interleaver.

5 Experimental results

We have run several experimental tests for different convolutional codes $\mathcal{C}$ and interleaver sizes N.
In the first test we used the convolutional code defined by the generator matrix in polynomial form $\mathcal{C}^1 = (1 + D + D^2 + D^5, 1 + D + D^3 + D^4 + D^6)$. This code satisfies one parity-check equation of weight 8. With a set of interleaved codewords we search for parity-check equations of weight 8 of $\mathcal{C}_{\pi}^1$ using a slightly improved method of [6] (we give in Table 2 the number M of

N	running time (in seconds)		
	$\mathcal{C}^1$	$\mathcal{C}^2$	$\mathcal{C}^3$
1 000	5	0.2	5
2 000	6	0.7	10
5 000	7	4	60
8 000	11	10	130
10 000	12	15	185

Table 1: Running time for $\mathcal{C}^1 = (1 + D + D^2 + D^5, 1 + D + D^3 + D^4 + D^6)$, $\mathcal{C}^2 = (1 + D + D^2, 1 + D^2 + D^3)$ and $\mathcal{C}^3 = (1 + D^2 + D^3 + D^5 + D^6, 1 + D + D^2 + D^3 + D^6)$

codewords that we use and the running time for recovering all parity-check equations), then we applied our method to reconstruct the interleaver and the convolutional code. For these tests we assumed that $s_{max} = 25$, and we give the running time in Table 1.

In the next test, the convolutional code was $\mathcal{C}^2 = (1 + D + D^2, 1 + D^2 + D^3)$. This code has 5 types of parity-check equation of weight 6. To test our method with $\mathcal{C}^2$ we assumed that $s_{max} = 10$, the running times are also in Table 1 and 2.

The last test was with the the convolutional code defined by $\mathcal{C}^3 = (1 + D^2 + D^3 + D^5 + D^6, 1 + D + D^2 + D^3 + D^6)$. This code satisfies 11 types of parity-check equations of weight 10. To reconstruct the interleaver and the convolutional code we assumed $s_{max} = 20$, see Table 1 and 2.

In all cases, the interleaver and the convolutional code were reconstructed efficiently. To obtain these running times we used all parity-check equations of weight 8, 6 or 10. Recovering all parity-check equations of low weight may take time, but our method can be applied without having all parity-check equations. For example, with the first convolutional code $\mathcal{C}^1 = (1 + D + D^2 + D^5, 1 + D + D^3 + D^4 + D^6)$, we note in Table 3 the running time for reconstructing the interleaver and the convolutional code in case we have less than 100% of parity-check equations of weight 8. We can see that, for small lengths the time increases rapidly if we do not have all parity-

N	$\mathcal{C}^1$		$\mathcal{C}^2$		$\mathcal{C}^3$	
	M	run. time	M	run. time	M	run. time
1 000	400	60	200	3	400	300
2 000	500	60	400	8	600	600
5 000	1400	600	900	45	1600	2000
8 000	2000	1800	1300	240	2400	3000
10 000	2600	2700	1700	300	2800	9000

Table 2: Running time in seconds for recovering all parity-check equations $\mathcal{C}^1 = (1 + D + D^2 + D^5, 1 + D + D^3 + D^4 + D^6)$, $\mathcal{C}^2 = (1 + D + D^2, 1 + D^2 + D^3)$ and $\mathcal{C}^3 = (1 + D^2 + D^3 + D^5 + D^6, 1 + D + D^2 + D^3 + D^6)$ without noise

check equations, but for large lengths having all parity-check equations is not necessary to reconstruct to reconstruct efficiently the interleaver and the convolutional code.

We also test with noisy interleaver codewords, for the convolutional code $\mathcal{C}^2$ and the binary symmetric channel of crossover probability $p = 0.001$ and $p = 0.01$, we note in table 4 the running time to recover almost all parity-check equations (more than 96% of them). These times are long but by parallelizing, the running time is divided by as much as executed programs. The running time to reconstruct the convolutional code and the interleaver is the same as in noiseless case.

6 Conclusion

This paper shows that when an interleaved convolutional code is used, then it can be efficiently reconstructed from the knowledge of a few hundred (or thousand) observed noisy codewords in the case of moderate noise of the channel by first recovering low-weight codewords in the dual of the interleaved convolutional code and then using this set of dual codewords to recover the convolutional structure and the interleaver. This assumption of moderate noise can be removed when the length N of the interleaver is sufficiently short (say below a few hundred) and is needed to ensure that most low-weight codewords are obtained by the slightly improved Cluzeau-Finiasz method [6]

N	% of parity-check equations	running time (in seconds)
1 000	100	5
	99	7
	96	37
	93	110
2 000	100	6
	95	13
	93	155
5 000	100	7
	95	78
8 000	100	11
	97	21
10 000	100	12
	94	63

Table 3: Running time for $\mathcal{C}^1 = (1 + D + D^2 + D^5, 1 + D + D^3 + D^4 + D^6)$

N	$p = 0.001$		$p = 0.01$	
	M	runn. time	M	runn. time
100	100	1	100	10
200	100	3	100	240
500	300	30	200	4 000
1 000	400	360	200	72 000
2 000	600	16 000		

Table 4: Running time in seconds for recovering all parity-check equations when $\mathcal{C} = (1 + D + D^2, 1 + D^2 + D^3)$ and with a crossover probability p

we used in our tests. Once these parity-check equations are recovered, a graph representing how these parity-check equations intersect is used to recover at the same time the interleaver and the convolutional code. This method is really fast, for instance the second phase took less than a few minutes in all our experiments and this even for very long interleavers (up to length $N = 10000$). This method applies to any convolutional code, it just needs convolutional codes that have reasonably low-weight and low-span codewords in the dual of the convolutional code, which is the case for virtually all convolutional codes used in practice.

References

[1] M. Marazin, R. Gautier, and G. Burel, "Algebraic method for blind recovery of punctured convolutional encoders from an erroneous bitstream," *IET Signal Processing*, vol. 6, no. 2, pp. 122–131, Apr. 2012.

[2] G. L. Rosen, "Examining coding structure and redundancy in DNA," *IEEE Engineering in Medecine and Biology*, vol. 25, no. 1, pp. 62–68, Jan. 2006.

[3] A. Valembois, "Detection and recognition of a binary linear code," *Discrete Applied Mathematics*, vol. 111, pp. 199–218, Jul. 2001.

[4] J. Barbier, G. Sicot, and S. Houcke, "Algebraic Approach of the Reconstruction of Linear and Convolutional Error Correcting Codes," in *World Academy of Science, Engineering and Technology*, vol. 16, Nov. 2006, pp. 66–71.

[5] M. Cluzeau and J.-P. Tillich, "On the Code Reverse Engineering Problem," in *Proc. of the IEEE Int. Symp. Information Theory*. Toronto, Canada: IEEE, 2008, pp. 634–638.

[6] M. Cluzeau and M. Finiasz, "Recovering a Code's Length and Synchronization from a Noisy Intercepted Bitstream," in *Proc. of the IEEE Int. Symp. Information Theory*. Seoul, Korea: IEEE, 2009, pp. 2737–2741.

[7] J. Zhou, Z. Huang, S. Su, and S. Yang, "Blind recognition of binary cyclic codes," *EURASIP Journal on Wireless Communications and Networking*, 2013.

[8] J. Wang, Y. Yue, and J. Yao, "A Method of Blind Recognition of Cyclic Code Generator Polynomial," in *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. IEEE, 2010.

[9] L. Wang, Y. Hu, S. Hao, and L. Qi, "The Method of Estimating the Length of Linear Cyclic Code Based on the Distribution of Code Weight," in *2nd International Conference on Information Science and Engineering (ICISE)*. IEEE, 2010, pp. 2459–2462.

[10] A.-D. Yardi, S. Vijayakumaran, and A. Kumar, "Blind reconstruction of binary cyclic codes," in *Proc of the European Wireless 2014*, 2014.

[11] B. Rice, "Determining the parameters of a rate $\frac{1}{n}$ convolutional encoder over $GF(q)$," in *Third International Conference on Finite Fields and Applications*, Glasgow, 1995.

[12] E. Filiol, "Reconstruction of Convolutional Encoders over $GF(q)$," in *Cryptography and Coding : 6th IMA Int. Conf.*, 1997, pp. 101–109.

[13] ——, "Reconstruction of punctured convolutional encoders," in *Int. Symp. on Information Theory and Applications (ISITA)*, 2000.

[14] P. Lu, L. Shen, X. Luo, and Y. Zou, "Blind Recognition of Punctured Convolutional Codes," in *Proc. of the IEEE Int. Symp. Information Theory*. Chicago, USA: IEEE, Jul. 2004, p. 457.

[15] J. Dingel and J. Hagenauer, "Parameter Estimation of a Convolutional Encoder from Noisy Observation," in *Proc. of the IEEE Int. Symp. Information Theory*. Nice, France: IEEE, Jun. 2007, pp. 1776–1780.

[16] F. Wang, Z. Huang, and Y. Zhou, "A Method for Blind Recognition of Convolution Code Based on Euclidean Algorithm," in *International Conference on Wireless Communications and Mobile Computing*. Shanghai: IEEE, Sep. 2007, pp. 1414–1417.

[17] M. Cluzeau and M. Finiasz, "Reconstruction of Punctured Convolutional Codes," in *Information Theory Workshop (ITW)*. Taormina, Italy: IEEE, Oct. 2009.

[18] M. Côte and N. Sendrier, "Reconstruction of convolutional codes from noisy observation," in *Proc. of the IEEE Int. Symp. Information Theory*. Seoul, Korea: IEEE, 2009, pp. 546–550.

[19] M. Marazin, R. Gautier, and G. Burel, "Dual code method for blind identification of convolutional encoder for cognitive radio receiver design," in *GLOBECOM Workshops*. Honolulu, USA: IEEE, 2009.

[20] M. Marazin, "Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en oeuvre d'un récepteur intelligent," Ph.D. dissertation, Université de Bretagne Occidentale, Dec. 2009.

[21] M. Marazin, R. Gautier, and G. Burel, "Some interesting dual-code properties of convolutional encoder for standards self-recognition," *Institution of Engineering and Technology Communications*, vol. 6, no. 8, pp. 931–935, May 2012.

[22] Y. Zrelli, M. Marazin, R. Gautier, and E. Rannou, "Blind Identification of Convolutional Encoder Parameters over GF(2^m) in the Noiseless Case," in *20th International Conference on Computer Communications and Neteworks (ICCCN)*. Maui, HI, USA: IEEE, 2011.

[23] Y. Zrelli, R. Gautier, M. Marazin, E. Rannou, and E. Radoi, "Focus on Theoretical Properties of Blind Convolutional Codes Identification Methods Based on Rank Criterion," in *9th International Conference on Communications*. Bucharest, Romania: IEEE, Jun. 2012, pp. 353–356.

[24] J. Zhou, Z. Huang, S. Su, and Y. Zhang, "Blind identification of convolutional codes in soft-decision situations," *International Journal of Modern Communication Technologies and Research (IJMCTR)*, vol. 2, Apr. 2014.

[25] S. Su, J. Zhou, Z. Huang, C. Liu, and Y. Zhang, "Blind identification of convolutional encoder parameters," *The Scientific World Journal*, vol. 2014, 2014.

[26] M. Bellard and J.-P. Tillich, "Detecting and reconstructing an unknown convolutional code by counting collisions," in *Proc. of the IEEE Int. Symp. Information Theory*. Honolulu, Hawaii, USA: IEEE, Jul. 2014, pp. 2967–2971.

[27] J. Barbier, "Reconstruction of turbo-code encoders," in *Proceedings of SPIE*, vol. 5819, 2005, pp. 463–473.

[28] ——, "Analyse de canaux de communication dans un contexte non coopératif," Ph.D. dissertation, École Polytechnique, Nov. 2007.

[29] M. Côte and N. Sendrier, "Reconstruction of a turbo-code interleaver from noisy observation," in *Proc. of the IEEE Int. Symp. Information Theory*. Austin, Texas, USA: IEEE, Jun. 2010, pp. 2003–2007.

[30] M. Cluzeau, M. Finiasz, and J.-P. Tillich, "Methods for the Reconstruction of Parallel Turbo Codes," in *Proc. of the IEEE Int. Symp. Information Theory*. Austin, Texas, USA: IEEE, Jun. 2010, pp. 2008–2012.

[31] A. Naseri, O. Azmoon, and S. Fazeli, "Blind Recognition Algorithm of Turbo Codes for Communication Intelligence Systems," *International Journal of Computer Science Issues*, vol. 8, no. 1, pp. 68–72, Nov. 2011.

[32] Y. Debessu, H. Wu, and H. Jiang, "Novel Blind Encoder Parameter Estimation for Turbo Codes," *Communications Letters*, vol. 16, no. 12, pp. 1917–1920, Dec. 2012.

[33] M. Marazin, R. Gautier, and G. Burel, "Blind Recovery of the Second Convolutional Encoder of a Turbo-Code when its Systematic Outputs are Punctured," *Military Technical Academy Review*, vol. XIX, no. 2, pp. 213–232, Jun. 2009.

[34] R. Gautier, M. Marazin, and G. Burel, "Blind Recovery of the Second Convolutional Encoder of a Turbo-Code when its Systematic Outputs are Punctured," in *7-th IEEE-Communications 2008*, Bucharest, Romania, 2008, pp. 345–348.

[35] P. Yu, J. Li, and H. Peng, "A least square method for parameter estimation of rsc sub-codes of turbo codes," *IEEE Communications Letters*, vol. 18, no. 4, Apr. 2014.

[36] D. Li and L. Guan, "A method of parameters estimation of sccc turbo code," in *Proc. of International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2013.

[37] J.-P. Tillich, A. Tixier, and N. Sendrier, "Recovering the interleaver of an unknown turbo-code," in *Proc. of the IEEE Int. Symp. Information Theory*. Honolulu, Hawaii, USA: IEEE, Jul. 2014, pp. 2784–2788.

[38] H. Lee, C. Park, J. Lee, and Y. Song, "Reconstruction of BCH Codes Using Probability Compensation," in *18th Asia-Pacific Conference Communications (APCC)*. Jeju, Island: IEEE, Oct. 2012, pp. 591–594.

[39] J. Wang, Y. Yue, and J. Yao, "Statistical Recognition Method of Binary BCH Code," *Communications and Network*, vol. 3, no. 1, pp. 17–22, Mar. 2011.

[40] J. Zhou, Z. Huang, C. Liu, S. Su, and Y. Zhang, "Information-Dispersion-Entropy-Based Blind Recognition of Binary BCH Codes in Soft Decision Situations," *Entropy*, vol. 15, no. 5, pp. 1705–1725, 2013.

[41] L. Wang and D. Li, "A new method for bch codes of blind recognition," *Advanced Materials Research*, vol. 631, pp. 1403–1408, 2013.

[42] H. Xie, F. Wang, and Z. Huang, "Blind recognition of reed-solomon codes based on histogram statistic of galois field spectra," *Advanced Materials Research*, vol. 791, pp. 2088–2091, 2013.

[43] W. Li, J. Lei, L. Wen, and B. Chen, "An Improved Method of Blind Recognition of RS Code Based on Matrix Transformation," in *Proc. of ICCT (International Conference on Communication Technology)*. IEEE, Nov. 2013.

[44] H. Zhang, H.-C. Wu, and H. Jiang, "Novel blind encoder identification of reed-solomon codes with low computational complexity," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 3294–3299.

[45] I. Kang, H. Lee, S. Park, J. Soh, and Y. Song, "Reconstruction Method for Reed-Muller Codes Using Fast Hadamard Transform," in *13th International Conference on Advanced Communication Technology (ICACT)*, Seoul, Korea, Feb. 2011, pp. 793–796.

[46] L. Lu, K. Li, and Y. Guan, "Blind Identification of Convolutional Interleaver Parameters," in *7th International Conference on Information,*

Communications and Signal Processing(ICICS). Macau, China: IEEE, Dec. 2009.

[47] L. Gan, D. Li, Z. Liu, and L. Li, "A low complexity algorithm of blind estimation of convolutional interleaver parameters," *Science China, Information Sciences*, vol. 56, no. 4, Apr. 2013.

[48] Y. Jia, L. Li, Y. Li, and L. Gan, "Blind Estimation of Convolutional Interleaver Parameters," in *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*. Shanghai, China: IEEE, Sep. 2012.

[49] Y.-Q. Jia, L.-P. Li, Y.-Z. Li, and L. Gan, "Blind estimation of communication emitter feature parameters," in *Proc. of International Conference on Computer and Information Technology (CIT)*. IEEE, 2012, pp. 281–285.

[50] J. Jeong, D. Yoon, J. Lee, and S. Choi, "Blind Reconstruction of a Helical Scan Interleaver," in *8th International Conference on Information Communications and Signal Processing (ICICS)*. Singapore: IEEE, Dec. 2011.

[51] H. Ryu, J. Lee, H. Hong, and D. Yoon, "Estimation of interleaver period for unknown signals," in *Proceedings of IC-NIDC (International Conference on Network Infrastructure and Digital Content)*. IEEE, 2010.

[52] G. Sicot and S. Houcke, "Blind detection of interleaver parameters," in *International Conference on Acoustics, Speech, and Signal Processing (ICCASP)*, vol. 3. Philadelphia, USA: IEEE, Mar. 2005, pp. 829–832.

[53] G. Sicot, S. Houcke, and J. Barbier, "Blind Detection of Interleaver Parameters," *Signal Processing*, vol. 89, no. 4, pp. 450–462, 2009.

[54] R. Gautier, G. Burel, M. Marazin, and C. Nsiala-Nzéza, "Blind Estimation of Block Interleaver Length and Encoder Parameters," *MTA Review*, vol. XXI, no. 1, pp. 31–44, Mar. 2011.